UFCEKG-20-2: Data, Schemas and Applications

Lecture 14
XML Validation and Schemas
Well formed XML (reminder xml lecture)

xml declaration (optional) used by xml processor; this document conforms to xml version 1 and uses the UTF-8 standard (Unicode optimized for ASCII)

root element; every well formed xml document must be enclosed by exactly one root element.

a comment; comments must be delimited by the <!-- --> characters as in xhtml

empty elements

attribute; attributes provide additional information about an element and consist of a name value pair; the value must be enclosed in a single (?) or double quote ("")

a simple element containing text

a complex element containing other elements and text
Well formed XML displayed in IE & Netscape

```
<xml version="1.0" encoding="UTF-8" >
  <patient xmlns="7503557856">
    <name>
      <first>Joseph</first>
      <middle>Michael</middle>
      <last>Bloggs</last>
    </name>
    <title>Mr</title>
    <address>
      <street>2 Gloucester Road</street>
      <city>Bristol</city>
      <postcode>BS2 4QS</postcode>
    </address>
    <tel>
      <home>0117 9541054</home>
      <mobile>07710 234674</mobile>
    </tel>
    <email>joe.bloggs@email.com</email>
    <fax/>
  </patient>
```

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<patient xmlns="7503557856">
  <name>
    <first>Joseph</first>
    <middle>Michael</middle>
    <last>Bloggs</last>
    <preferred>Joe</preferred>
  </name>
  <title>Mr</title>
  <address>
    <street>2 Gloucester Road</street>
    <city>Bristol</city>
    <postcode>BS2 4QS</postcode>
  </address>
  <tel>
    <home>0117 9541054</home>
    <mobile>07710 234674</mobile>
  </tel>
  <email>joe.bloggs@email.com</email>
  <fax/>
</patient>
```
Vocabularies and Validity

- XML documents are not directly written; instead XML is used to create one or more vocabularies, specific custom markup languages (often referred to as XML applications), and it is these languages which are used to create documents.

- such a language (a set of namespaces, elements, attributes etc. – a vocabulary) is defined using a set of rules which specify the set (potentially infinite) of complying documents.

- such a set of rules is generically referred to as a schema.

- for instance, in our example document, we may want to specify rules that state that the <name> element must always contain exactly one each of the <first>, <middle>, <last>, <previous> & <preferred> elements and that they must occur in this order.

- additional rules we might want to specify are that the <first> & <last> elements must always contain alphanumerics (not empty) and that they must never exceed 256 characters each.
So what again is Validation?

A document conforming to a particular schema is said to be valid, and the process of checking that conformance is called validation.

Schema languages differentiate between at least four levels of validation:

- The validation of the markup -- controlling the structure of a document.

- The validation of the content of individual leaf nodes (datatyping).

- The validation of integrity, i.e. of the links between nodes within a document or between documents.

- Any other tests (often called "business rules").
XML schema systems (1)

- more formally, an XML schema language is a *formalization* of the *constraints*, expressed as *rules* or a *model* of structure, that apply to a *class* of XML documents.

- an XML document constrained (described) by a schema is called an *instance* document and such a document is considered *schema-valid*.

- schemas can serve as design tools, establishing a framework on which *implementations* can be built.

- many schema languages are now available including DTD, [W3C Schema](http://www.w3.org/XML/schemas), [Schematron](http://www.schematron.org/), [NG Relax](http://www.relaxng.org/), [Examplotron](http://www.schematron.org/) and others.

- the most widely used of these is W3C Schema (XSD) but first we briefly consider the Document Type Definition (DTD) approach which originated in the days of SGML.
XML Schema Systems (2)

- A set of rules that are used to validate a XML document is referred to as a *schema*. A document conforming to a particular schema is said to be *valid* against that *schema*, and the process of checking that conformance is called *validation*.

Schema languages differentiate between at least four levels of validation:

- The validation of the *markup* -- controlling the structure of a document.
- The validation of the content of individual leaf nodes *(data-typing)*
- The validation of *integrity*, i.e. of the links between nodes within a document or between documents.
- Any other tests (often called "*business rules"."

There are currently two types of schema languages:

- **grammar based** - for specifying structure, form, and syntax (e.g. DTD, XML Schema, Relax NG)
- **rule based** - for expressing data relationships, such as operational and business rules (e.g. Schematron)
XML schema systems (3)

The Document Type Definition (DTD) approach.

- DTD’s are written in a formal notation (BNF) that specifies exactly which elements and entities may appear where in the document and what the elements’ contents and attributes are.

- A DTD can make statements of the type such as “a ‘ul’ element can only contain ‘li’ elements” and every ‘student’ element must have a ‘student_number’ attribute”

- Hence a DTD lists all the elements, attributes and entities the document uses and the context in which it uses them.

- A validating parser compares a document to its DTD and lists any places where the document differs from the DTD.

- Validity operates on the principal that everything not permitted is forbidden.

- If an instance document satisfies the DTD it is said to be valid otherwise it is said to be invalid.
XML schema languages (1)

Example DTD for patient.xml

```xml
<!ELEMENT patient (name, title, address, tel+, email?, fax?)>
<!ATTLIST patient
    nhs-no CDATA #REQUIRED>
<!ELEMENT name (first, middle, last, previous, preferred)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT address (street+, city, county, postcode)>
<!ELEMENT tel (home*, mobile*)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT fax (#PCDATA)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT middle (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT previous (#PCDATA)>
<!ELEMENT preferred (#PCDATA)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT county (#PCDATA)>
<!ELEMENT postcode (#PCDATA)>
<!ELEMENT home (#PCDATA)>
<!ELEMENT mobile (#PCDATA)>
```
XML schema languages (2)

So what’s the problem with DTD’s?

- DTD’s work (to an extent) but there are many issues and limitations with this approach, for example DTD’s do not specify
  - how many instances of each kind of element appear in a document
  - what the character data inside the element look like
  - the semantic meaning of the element; for instance, whether it contains a date or a person’s name.
  - DTD’s cannot specify anything about the length, structure, meaning, allowed values, or other aspects of the text content of an element.
  - what the root element of a document is
- DTD’s are not in themselves XML documents
XML schema languages (3)

W3C XML Schema

- XML Schemas ([http://www.w3.org/XML/Schema](http://www.w3.org/XML/Schema)) offers a much more powerful way of constraining XML documents than DTD’s.

- Advantages of Schemas over DTD’s include:
  - in addition to the traditional constraints, XML Schemas allow content model constraints for generic data formats to be built.
  - these defined constraints can be shared (using namespaces) and referenced from other schemas using [XLink](http://www.w3.org/TR/xlink) and [XPointer](http://www.w3.org/TR/xpointer).
  - it follows an object oriented approach, allowing for the definitions of *types* and *inheritance* which allows for better maintainability and can save a significant amount of design time.
XML schema languages (4)

W3C XML Schema: simple example

- consider the following simple document

```xml
<?xml version="1.0"?>
<studentName>Joseph Bloggs</studentName>
```

- assuming that the studentName element can only contain a simple string value, the schema for this document would look like

```xml
<?xml version="1.0"?>
<xs:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema>
  <xs:element name="studentName" type="xs:string" />
</xs:schema>
```

- Validating an instance doc against its schema requires a validating parser such as the Xerces parser from the Apache XML Project (http://xml.apache.org/xerces-j/)
W3C XML Schema: simple and complex types

- schemas support two different types of content: *simple* and *complex*. Simple types equates with basic data types (strings, integers, dates, times, etc.) – simple types by definition cannot contain nested elements.

```xml
<xs:element name="studentName" type="xs:string"/>
```

- elements that complex types may contain nested elements and attributes. Only elements can have complex types, attributes always have simple types.

```xml
<xs:complexType name="addressType">
  <xs:sequence>
    <xs:element ref="street" minOccurs="2" maxOccurs="unbounded"/>
    <xs:element ref="city"/>
    <xs:element ref="county"/>
    <xs:element ref="postcode"/>
  </xs:sequence>
</xs:complexType>
```
XML schema systems (6)

W3C XML Schema: local versus global declarations

- Instance elements declared at the top level of the schema (immediate child of the xs:schema element) are considered global elements. According to the schema specification, any elements declared globally can act as the root element of the instance doc.

- Elements declared with another element declaration (i.e. within a complex type) are considered local. You can element declarations within a schema that have the same name but different semantics if they are declared locally.

- The side effect of using global declarations may include:
  - Naming conflicts when schemas are shared and/or merged
  - If more than one element is declared globally, a schema valid document may not contain the expected root element
XML schema systems (7)

W3C XML Schema: attributes, data-types and derivation

- attribute declarations
  - attributes are declared using the xs:attribute element. Attributes may be declared globally or locally as part of a complex type definition.

- data-types
  - there are great range of data-types built into XML Schema; xs:string, xs:integer, xs:dateTime, xs:decimal etc. etc.

- derivation
  - there are three derivation methods in XML Schema
  - derivation by restriction – where constraints are added on datatype without changing its original meaning,
  - derivation by list – where new data-types are defined as being lists of values belonging to a data type
  - derivation by union – where new data-types are defined as allowing values from a set of other data types and lose most of their meaning
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type ="text/xsl" href='patient.xslt'?>
<patient xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="patient.xsd">
  <name>
    <first>Joseph</first>
    <middle>Michael</middle>
    <last>Bloggs</last>
    <previous/>
    <preferred>Joe</preferred>
  </name>
  <title>Mr</title>
  <address>
    <street>2 Gloucester Road</street>
    <city>Bristol</city>
    <county>Avon</county>
    <postcode>BS2 4QS</postcode>
  </address>
  <tel>
    <home>0117 9541054</home>
    <mobile>07710 234674</mobile>
  </tel>
  <email>joe.bloggs@email.com</email>
  <fax></fax>
</patient>

<xs:element name="patient">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="nameType"/>
      <xs:element name="title" type="titleType"/>
      <xs:element name="address" type="addressType"/>
      <xs:element name="tel" type="telType" maxOccurs="2"/>
      <xs:element name="email" type="emailType" minOccurs="0"/>
      <xs:element name="fax" type="xs:string" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="nhs-no" type="xs:integer" use="required"/>
  </xs:complexType>
</xs:element>

<xs:complexType name="nameType">
  <xs:sequence>
    <xs:element name="first" type="nameStringType"/>
    <xs:element name="middle" type="nameStringType"/>
    <xs:element name="last" type="nameStringType"/>
    <xs:element name="previous" type="nameStringType"/>
    <xs:element name="preferred" type="nameStringType"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="nameStringType">
  <xs:restriction base="xs:string">
    <xs:maxLength value="64"/>
  </xs:restriction>
</xs:simpleType>
Resources

Introduction to XML Schemas

W3C Schools XML Schema Tutorial
http://www.w3schools.com/Schema/default.asp

On-line Schema Validator
http://schneegans.de/sv/